Notes on creating web apps with google!

Introduction

Ok let's get down to it.

I’m going to begin with what I know how to do, and hopefully as I learn I can update this document and eventually it will be something I can use later to remember, refresh and keep building web stuff.

WEB APPS!

A web app. Jumping straight to the big guns while skipping the small fry. A very ME thing to do, that’s for sure. Right into the deep end.

What’s a web app? Ok, I was going to say easy. But it’s not. A web app will take information from the user, bring it down to some machine server, RUN SOME FUCKING CODE ON IT, and respond.

The kicker? The code being run can be fucking ANYTHING like c++ or Python.

What do you need for it to work?

1. A fucking server to host the website and data. And a domain name.
2. A fucking machine server to run the code that will then upload to the server from A.
3. A fucking apache or sql type database to store information and retrieve information in order for the web app to function.
4. SOME FUCKING WAY TO CONNECT IT ALL.

Ain’t nobody go time fo’ that.

Google makes it easy, as it always does.

Google web apps. Do B, C and D all for free.

## Google apps and Java

How do you make UI without direct HTML code?  
The answer is to use a **Template System.** You can use any template system you want, but for now, we will stick with *Django* with *Java Servlet Page(JSP)* holding the internal logic.

Django Template System

*You can read the specifics online, but essentially, it allows you to make HTML, but it’s more powerful. You can have inheritance, forms, tags that do cool things, etc. Template systems don’t even need to be specifically for HTML, they can template ANYTHING, Java, C++, etc, etc. It’s just a means of abstracting code, making it even more generalized. You can even do inheritance!*

Java Servelt Page

*JSP is a text document that holds two things: Static elements and JSP elements.  
- The static elements are just things like HTMl, CSS, PHP, etc.  
- The JSP elements are the dynamic elements that intermingle with the static elements to produce unique pages.   
So you would have something like  
<html>… JSP ELEMENT …</html>  
The JSP element is dynamic and can be anything, for instance an entire java class system to animate birds.   
Read more on JSP’s on my note page for JSP’s.*

In Google apps, any JSP page in /war is AUTOMATICALLY converted to a unique html page /<JSP FILENAME>.

Requests and Responses in JSP and beyond

A "request" in server-side processing refers to the transaction between a browser and the server.  When someone clicks or enters a URL, the browser sends a "request" to the server for that URL, and shows the data returned.  As a part of this "request", various data is available, including the file the browser wants from the server, and if the request is coming from pressing a SUBMIT button, the information the user has entered in the form fields.

The JSP "request" variable is used to obtain information from the request as sent by the browser.  For instance, you can find out the name of the client's host (if available, otherwise the IP address will be returned.)  Let us modify the code as shown:

<HTML>

<BODY>

<%

    // This scriptlet declares and initializes "date"

    System.out.println( "Evaluating date now" );

    java.util.Date date = new java.util.Date();

%>

Hello!  The time is now

<%

    out.println( date );

    out.println( "<BR>Your machine's address is " );

    out.println( request.getRemoteHost());

%>

</BODY>

</HTML>

A similar variable is "response".  This can be used to affect the response being sent to the browser.  For instance, you can call response.sendRedirect( anotherUrl ); to send a response to the browser that it should load a different URL.  This response will actualy go all the way to the browser.  The browser will then send a different request, to "anotherUrl".  This is a little different from some other JSP mechanisms we will come across, for including another page or forwarding the browser to another page.